IERG 4210
Web Programming & Security
Tutorial 7

Fan YANG
(Modified from the slides of former TA Menghan Sun)

Previous Phase

- Main Website
  - Phase 2 - Content dynamically from database
  - Phase 3 - AJAX shopping list
- Admin panel (admin.html/admin.php, admin-process.php)
  - Phase 2 - Maintain the product database

Phase 4: Secure your website

- Prevent XSS, CSRF, SQL attacks (Phase 4.1-4.3, 4.5) -> Next Tutorial
- Authentication for Admin Panel (Phase 4.4, 4.5) -> today
  - Otherwise everyone can manipulate your database.
- Apply SSL certificate (Phase 4.6) -> today
  - Do it first, it takes time to apply

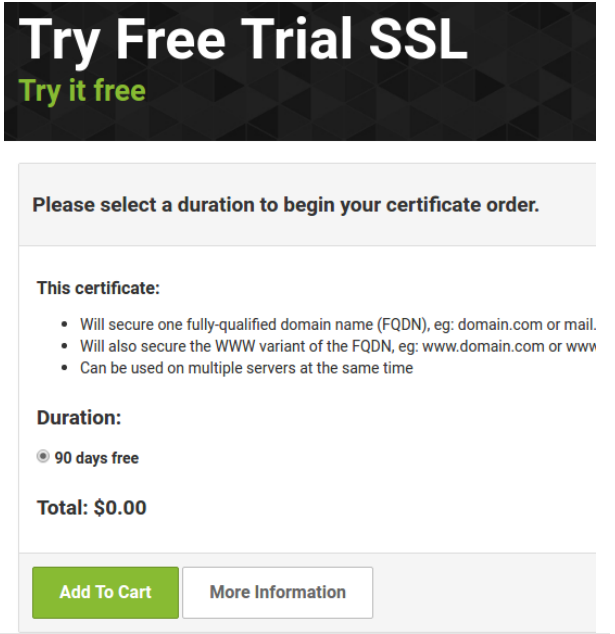
Apply TLS/SSL to your website

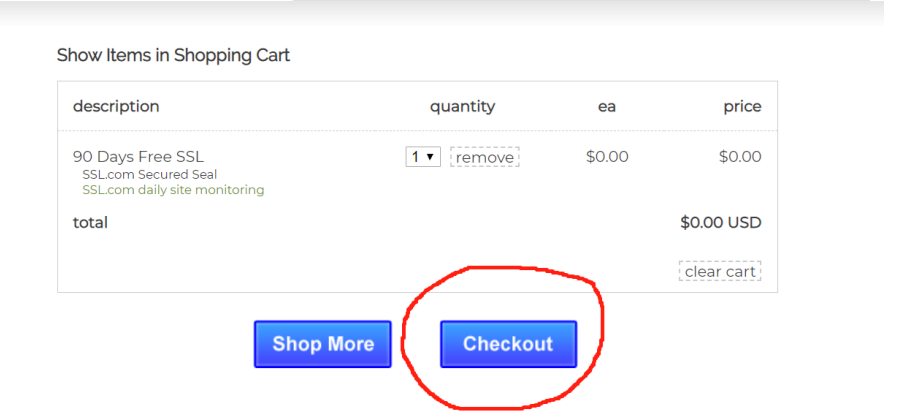
Replace it with your number here!

- ssh to your server
- In your shell, input following commands
- `openssl req -nodes -newkey rsa:2048 -keyout secure.s1.ierg4210.ie.cuhk.edu.hk.key -out server.csr`
- In the interactive prompt:
  - Country Name (2 letter code) [XX]:HK
  - State or Province Name (full name) []:Hong Kong
  - Locality Name (eg, city) [Default City]:
  - Organization Name (eg, company) [Default Company Ltd]:CUHK
  - Organizational Unit Name (eg, section) []:
  - Common Name (eg, your name or your server's hostname) []:secure.s81.ierg4210.ie.cuhk.edu.hk
  - Email Address []:your email
- DO NOT input password at this step or your apache can not read it!
- **Just put the crt and key file in somewhere inaccessible by common users**

Sign up for free

- https://www.ssl.com/certificates/free/buy is illustrated here
- Open server.csr you created and paste into the field
  - Start with
    -----BEGIN CERTIFICATE REQUEST-----
  - Choose Apache-MOD SSL
- After ~10 mins, you will get the certificate via an email





Order successfully placed. Click here to finish processing your ssl.com certificates. X close

Show Order Transaction

Click here to finish processing this certificate order.

Order Details

reference number: c1d3-1dufn4r
date of order: 2018-11-11 01:39:07 -0600
payment method: n/a (free)

	DESCRIPTION	QUANTITY	PRICE
1	90 Days Free SSL SSL.com Secured Seal SSL.com daily site monitoring co-af1dufn4r	1	$0.00
		Subtotal:	$0.00
		Total:	$0.00

Tags

SSL Certificates Management

Subject	Folder	Status	Order Date	Expires	Action
credit - free certificate open : finish processing		waiting for csr	Nov 11, 2018	n/a	submit csr

10

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIICizCCAkkCAQAwgYUxCzAJBgNVBAYTAlVTMQ4wDAYDVQQIEwVUZXhhcz
EQMA4GA1UEBxMHSG91c3RvbjEpMCcGA1UEChMgU2VjdXJlIFNvY2tldHMg
TGFib3JhdG9yaWVzLCBMTEMxEDAOBgNVBAsTB0V4YW1wbGUxFzAVBgNVBA
(more encoding...)
gsPa9X3F8h82UJDqg9bZ+fPQely1JeEg1sVJW2sTaRNnAqsddPFihWHfzS
/+5Ks5E/MYsKBV4GjjytxR3PgxWT06XFqnEt40RDTJ2wd7kQ9yGYpFpQhw
92SQTgt3fbz4TmbTmDU7fYw1lq2kQLx6z6oAAwCwYHKoZIzjgEAwUAAy8A
V7VB4cjEAhRFFKjEGjuJcPUSXUMB85RYyGpSrw==
-----END NEW CERTIFICATE REQUEST-----
```


Note: The Common Name (CN) field in your CSR must conform to the following rules:

- It MUST represent your fully-qualified domain name (i.e. submit.domain.com)
- It CANNOT not represent an ip address (i.e. 58.123.123.123 or 127.0.0.1 - see deprecated names)

*CSR:
certificate signing request

Save to CSR Manager: ☑

Managed CSR: none Go To CSR Manager

Server Software: OTHER
for informational purposes only

How can we he


```
[ec2-user@ip-172-31-22-22 ~]$ cat server.csr
-----BEGIN CERTIFICATE REQUEST-----
MIIC3TCCAcUCAQAwgZcxCzAJBgNVBAYTAkhLMRIwEAYDVQQIDAlIb25nIEtvbmcx
FTATBgNVBAcMDERlZmF1bHQgQ2l0eTENMAsGA1UECgwEQ1VISzEqMCgGA1UEAwwh
c2VjdXJlLnQyLmllcmc0MjEwLmllLmN1aGsuZWR1LmhrMSIwIAYJKoZIhvcNAQkB
FhNkc2s5MzA4MTdAZ21haWwuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEAxmGyWPU7fEYnUb4DGtz+JidUlNt1EJ9f9mXoDBWqZgFU4vliV2DcrgnB
F2OrhHOaoFnNrPr9raJxLn1rTcyoxm15cELXYKH1xwAWZC8EJI+wId7ky4LNwKI0
CLaIrVwTqnqexNAPpYuVkL5A+oHempbrgfK12zQ1swVaqasn2bsO5OxnzXyCinNH
eXm+sxF/wmcxRbL3860I29dRM2qg3LIANizrNlGA/q1PCB+2TxgaMYlnPlLWkBgJ
6KV7z3SxG4Y4zhgmRXkWf/MNjp4J596Z1hhLj4TjU0AZJElqyxxkms2newQTyCYr
+SanPbRFzf0rP7mWTZbyU67rwBJ6sQIDAQABoAAwDQYJKoZIhvcNAQELBQADggEB
ACz4PcVo1aDJWsVIEl4k/9Hn1RJoDFnYB0N9ZES+tHeJA9un1GnDv75JGsW+py9g
BPTl5GepdOjGDjCzf0x+q60fKwGIrlGmT+SGBQYm07x8gxrDZF8U32p+kLDMnfcK
XNk8EXcWu/5jXRZyjVndYpUuAkPzsPCNeAXK7qH6GBekzDguQqU+ol/0GIQD5+WK
G82klLcd2vLi656mLCUszxb0hxOVoYwMb0fkoqM2V+acbsBevWK0u3DUrfEDpPnR
FwsSS4oKVTrIP3hOd/7n3SWK6Xtd403QZACrr4JCJGJ6evPK3gjXRMtGKlLigiMF
1jJ86HMInbRoPIh5oh73ZRQ=
-----END CERTIFICATE REQUEST-----
[ec2-user@ip-172-31-22-22 ~]$
```

✓1 Submit CSR ✓2 Registrant 3 Perform Validation 4 Complete

Create Contacts For secure.t2.ierg4210.ie.cuhk.edu.hk (order ref# co-af1dufn4r)

Please provide point of contacts for secure.t2.ierg4210.ie.cuhk.edu.hk.

+ Create Contact

Roles

Administrative ☑
Billing ☐
Technical ☐
Validation ☐

Details

*First Name: Shuaike

*Last Name: DONG

Selected Contacts (1)

	Administrative	Billing	Technical	Validation	
Dong, Shawn |	☑	☐	☐	☐	Remove

Available Contacts

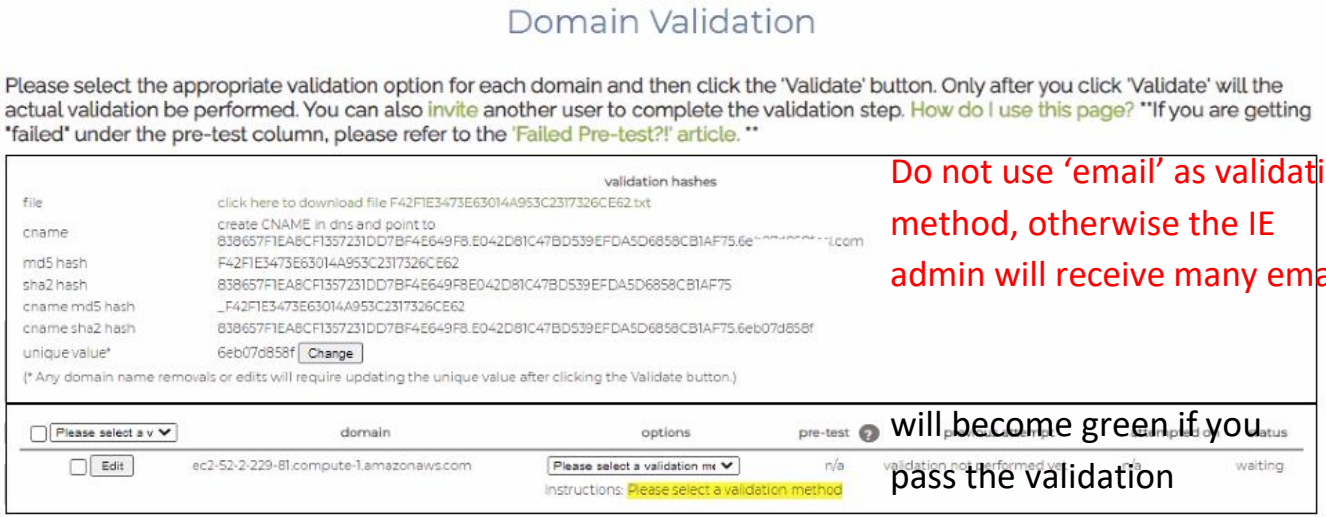
Administrative	Billing	Technical	Validation

*Note: updating role of "Available Contacts" contact will update roles for other certificates where this contact was also used.

Cancel Next >>

Validation

- `sudo mkdir -p /var/www/html/.well-known/pki-validation`
- Follow the instruction to download the .txt file
- `sudo cp xxx.txt /var/www/html/.well-known/pki-validation`
- Select “CSR hash text file using http://”, Click “Validation”



Do not use ‘email’ as validation method, otherwise the IE admin will receive many emails

will become green if you pass the validation

Download the CRL File

SSL Certificate For secure.t2.ierg4210.ie.cuhk.edu.hk [how do I use this page?]

Subject	Status	Order Date	Expires	Action
secure.t2.ierg4210.ie.cuhk.edu.hk open : download : documents : seal	issued	Nov 11, 2018	Feb 09, 2019	renew or change domain(s)/rekey

certificate details		validation status		smart seal	
certificate type	duration	validation level	certificate #	issued on	requested on
Free	90 days	Class 1 DoD	co-af1dufn4r	Nov 11, 2018	Nov 11, 2018

certificate contents
algorithm: SHA2

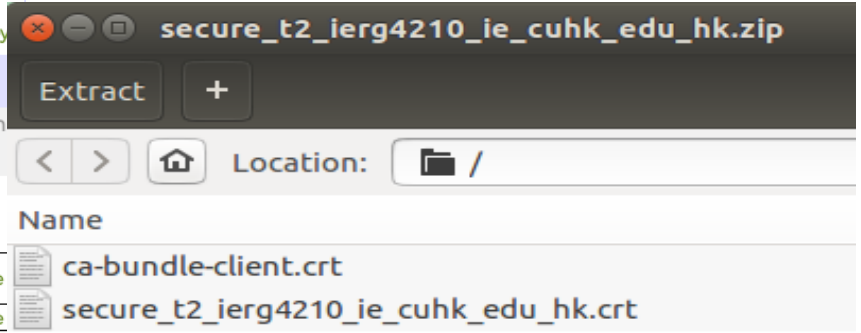
verify and troubleshoot
check ssl installation
visit site with ssl
visit site without ssl

for developers
preformatted api strings
developer tools

registrant
CUHK
CUHK
Default City, Hong Kong,
000000
US
edit registrant

certificate download by platform

Microsoft IIS (*.p7b)	download	guide
WHM/cpanel	download	guide
Apache	download	guide
Amazon	download	guide
Nginx	download	guide
V8+Node.js	download	guide
Java/Tomcat	download	guide
Other platforms	download	guide
CA bundle (intermediate certs)	download	guide



Download
Apache file

```
scp -i ierg4210_2018.pem
secure_t2_ierg4210_ie_cuhk_edu_hk.crt ca-bundle-
client.crt ec2-user@[your ip]:~/

sudo cp ~/secure_t2_ierg410_ie_cuhk_edu_hk.crt
~/ca-bundle-client.crt /var/www/

sudo cp ~/secure.t2.ierg4210.ie.cuhk.edu.hk.key
/var/www
```

First download the ziped certificates from SSL.com

Upload them via SCP/FileZilla/WinSCP to your server

copy them to /var/www/

Install the Server Certificate

This tutorial is based on Amazon Linux 2 AMI, if you are using Linux AMI, type this instead:

```
sudo yum install
mod24_ssl
```

Common error from students:
NOT httpd.conf !!!

- Install mod_ssl in your server via `sudo yum install mod_ssl`
- Modify the SSL configuration file
  - `sudo vim /etc/httpd/conf.d/ssl.conf`
  - Copy the server certificate file to /var/www and modify the value of SSLCertificateFile Item
    - `SSLCertificateFile /var/www/secure_s1_ierg4210_ie_cuhk_edu_hk.crt`
  - Copy the private key file and modify
    - `SSLCertificateKeyFile /var/www/secure.s1.ierg4210.ie.cuhk.edu.hk.key`
  - Copy the server certificate chain file and modify
    - `SSLCertificateChainFile /var/www/ca-bundle-client.crt`

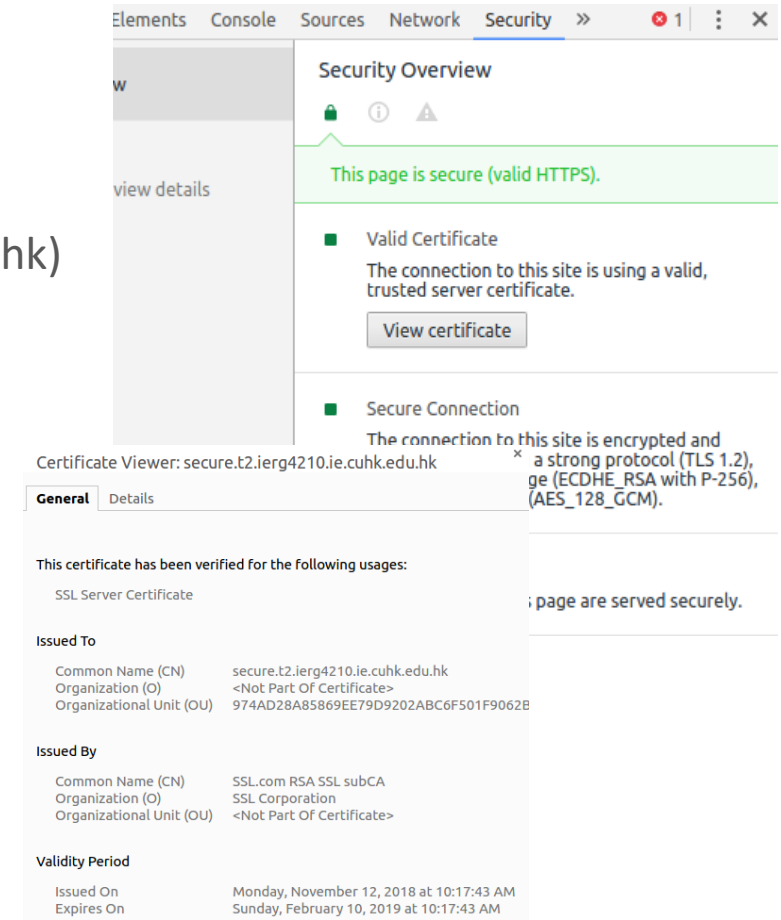
```
#   Server Certificate:
# Point SSLCertificateFile at a PEM encoded certificate.  If
# the certificate is encrypted, then you will be prompted for a
# pass phrase.  Note that a kill -HUP will prompt again.  A new
# certificate can be generated using the genkey(1) command.
SSLCertificateFile /var/www/secure_t2_ierg4210_ie_cuhk_edu_hk.crt

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile /var/www/secure.t2.ierg4210.ie.cuhk.edu.hk.key

#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
#   concatenation of PEM encoded CA certificates which form the
#   certificate chain for the server certificate. Alternatively
#   the referenced file can be the same as SSLCertificateFile
#   when the CA certificates are directly appended to the server
#   certificate for convinience.
SSLCertificateChainFile /var/www/ca-bundle-client.crt
```

Install the Server Certificate (cont.)

- Make sure user apache has privilege of reading 3 files in last step
  - `sudo chgrp apache xxx.crt xxx.key xxx.crt`
- Open port 443 in AWS security group!
- Restart the apache server
  - `sudo systemctl restart httpd`

Check the certificate

- Visit using the browser your website (https://secure.s1.ierg4210.ie.cuhk.edu.hk)
- If you use Chrome
  - Developer Tool (F12)
  - Go to Security tab
  - View certificate



Redirect HTTP requests to HTTPS

- You can redirect user if they access http://secure... or http://..../admin.php
- Reference: https://wiki.apache.org/httpd/RedirectSSL
- Using virtual hosts:
  - create a .conf file and include it in /etc/httpd/conf/httpd.conf
  - restart httpd

```
<VirtualHost *:80>
   ServerName s76.ierg4210.ie.cuhk.edu.hk
   Redirect / https://secure.s76.ierg4210.ie.cuhk.edu.hk/
</VirtualHost>
```

- Using .htaccess file

http://s76.ierg4210.ie.cuhk.edu.hk -> https://secure.s76.ierg4210.ie.cuhk.edu.hk

Authentication for Admin Panel – Phase 4.4

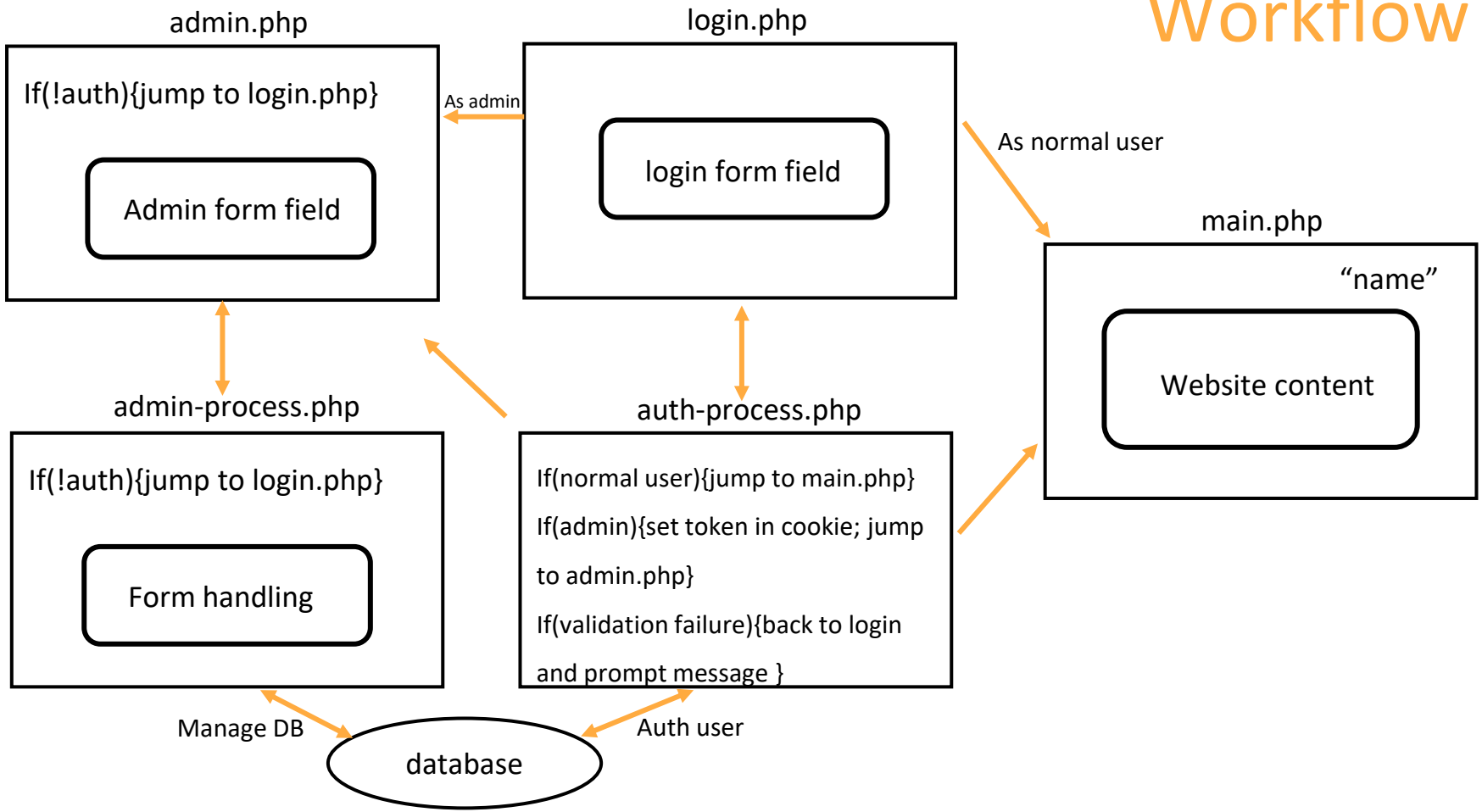
- A website page and an admin page. But everyone can access admin panel now.
  - We need to add an admin user to the user management database

    (only user with special privilege could visit admin page and do operations)
  - Store hashed passwords in database (user table) (// Why not original?)
  - Build a login page and perform the authentication.
  - Use cookies to remember the authentication result. (via maintaining the token)
  - Support logout and password changing

Phase 4.4

- Create a user table
- Login.php
- Maintain an authentication token using Cookies
- Validate the token
- Support logout and password changing

Workflow


admin.php
If(!auth){jump to login.php}
Admin form field
As admin
login.php
login form field
As normal user
main.php
“name”
Website content
admin-process.php
If(!auth){jump to login.php}
Form handling
auth-process.php
If(normal user){jump to main.php}
If(admin){set token in cookie; jump to admin.php}
If(validation failure){back to login and prompt message }
Manage DB
database
Auth user

Phase 4.4

- ❑ Create a user table
- ❑ Login.php
- ❑ Maintain an authentication token using Cookies
- ❑ Validate the token
- ❑ Support logout and password changing

Hash Function

- Accept variable size message M and produce a fixed-size digest h(M)
  - h(M) can be thought as "fingerprint" of M
- A "good" hash function:
  - Easy to compute h(M)
  - Computationally infeasible to find M from h(M)
  - Computationally infeasible to find collision ($X \neq Y$, but h(X)=h(Y))

    However, collision always exists since the length of messages is longer than that of digest.
- Secure Hash Functions
  - Offline-dictionary attack: pre-computed a list of hashed values to create a lookup table
  - Salting, i.e., add a random string to expand the effective space for brute-force attack
  - Many hash functions, some are broken: MD5, SHA-1,…
  - Just call the existing libraries; don't implement the algorithm yourself

Database – User Table

- Create user table to save userid (primary key), email, salt, "salted **and** hashed password", admin flag.
  - flag (e.g., integer 1) to indicate "admin" or not
- Every user has its own random salt, so the salted password generated by below will be different

```
<?php
echo ($salt = mt_rand()).“<br/>”;
echo hash_hmac('sha256', $_REQUEST['password'], $salt);
?>
```

Message Authentication Code built from hash

Secret key for HMAC

- Adding a user: `INSERT INTO account (email, salt, password) VALUES`

```
("1@gmail.com","1160029811","5d2b3d93eba5eb05e34b7c2301c517a17c593bc364ca88fa3417944cb5a4e74d");
```

More “Advanced” Discussion

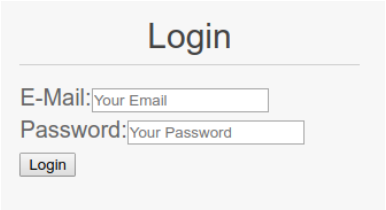
- Hashing password is more complicated than you think
  - What we show in the last slide is the “simplest” thing you can do
  - (at least better than not using salt, using a short salt, or using the same salt for everyone)
- Many things you can do it better:
  - you may even want to store the salt in another server (need to compromise 2 servers then)
  - mt_rand() does not generate “cryptographicly secure values”
  - i.e., use random_int(), random_bytes(), or openssl_random_pseudo_bytes() instead
  - you want the attacker’s trial to be slow while your normal operation reasonably fast
  - i.e., apply another layer of bcrypt() function
- e.g., see https://wiki.mozilla.org/WebAppSec/Secure_Coding_Guidelines#Password_Storage

Phase 4.4

- ❑ Create a user table
- ❑ Login.php
- ❑ Maintain an authentication token using Cookies
- ❑ Validate the token
- ❑ Support logout and password changing

https://secure.s76.ierg4210.ie.cuhk.edu.hk/login.php

Login Page



- Build login.php and auth-process.php
  - Create the HTML yourself
  - Form will be submitted to auth-process.php
    - submit email, password (first validate the format using *preg_match*)
    - get “salt” from DB, compute the “salted hash value” then compare.
    - lead admin to admin panel, common user to main page, refuse incorrect password.
      - header(‘Location: xxx’);
- Now you have:
  - login, admin, mainpage
  - Related process file auth-process, admin-process
  - Every time need password?
    - Set admin token kept in cookie.

Sample Code

```php
function ierg4210_login(){
  if (empty($_POST['email']) || empty($_POST['pw'])
    || !preg_match("/^[\w=+\-\/][\w='+\-\/\.]*@[\w\-]+(\.[\w\-]+)*(\.[\w]{2,6})$/", $_POST['email'])
    || !preg_match("/^[\w@#$%\^\&\*\-]+$/", $_POST['pw']))
    throw new Exception('Wrong Credentials');

  // Implement the login logic here


  if ($login_success){
    // redirect to admin page
    header('Location: admin.php', true, 302);
    exit();
  } else
    throw new Exception('Wrong Credentials');
}


function ierg4210_logout(){
  // clear the cookies and session

  // redirect to login page after logout
  header('Location: login.php',true,302);
  exit();
}
```

Phase 4.4

- ❑ Create a user table
- ❑ Login.php
- ❑ Maintain an authentication token using Cookies
- ❑ Validate the token
- ❑ Support logout and password changing

Review: Cookie and PHP Session

- Cookie has been discussed extensively in Lecture 7
- PHP Session
  - Cookie -> stored in client; $_SESSION -> stored in server (temp files)
  - To use: session_start(), $_SESSION
  - How? You only use a variable $_SESSION, how do you know which user ?
  - Based on a random PHPSESSID: in essence, is a cookie or GET parameter

```php
<?php
session_start();
$_SESSION['username'] = 'niki';
?>
```

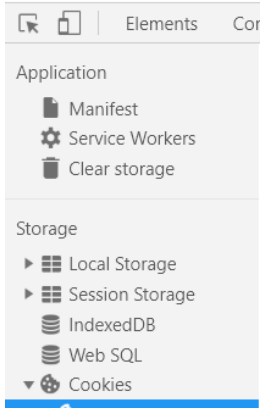
pageA.php

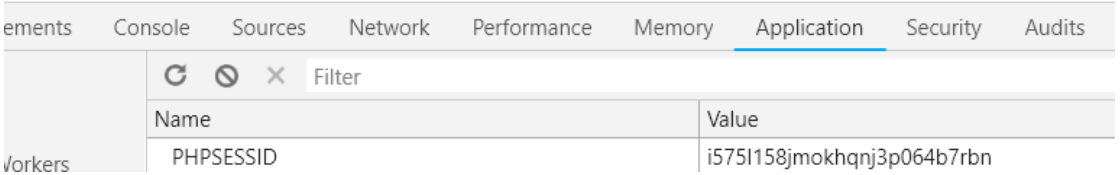
```php
<?php
session_start();
echo $_SESSION['username']; ?>
```

pageB.php

Review: Cookie and PHP Session

- PHP Session
  - To use: session_start(), $_SESSION
  - PHPSESSID: in essence, is a cookie or GET parameter
    - (You can inspect the cookie with browser developer tool)
    - So the server know who you are, then fetch your $_SESSION[ ] according to PHPSESSID, open different storage
    - By default, it expires when the browser is closed





Maintain an authentication token using cookie

- Setting Cookie when validate successfully.
- Refer to lecture notes (how to set cookie):

```
$q=$db->prepare('SELECT * FROM account WHERE email = ?');
$q->execute(array($email));
if($r=$q->fetch()){
    //Check if the hash of the password equals the one saved in database
    //If yes, create authentication information in cookies and session
    //program code on next slide
}
```

Maintain an authentication token using Cookie

```
$saltedPwd = hash_hmac('sha256', $pwd, $r['salt']);
if($saltedPwd == $r['password']){
  $exp = time() + 3600 * 24 * 3;  //3days
  $token = array(
    'em'=>$r['email'],
    'exp'=>$exp,
    'k'=>hash_hmac('sha256', $exp.$r['password'], $r['salt'])
    );
  //create the cookie
  setcookie('s4210', json_encode($token), $exp,'','',true,true);
  //put it also in session
  $_SESSION['s4210'] = $token;
  return true;
}
```

Set token

The token shouldn’t reveal original password

Server side session

Secure, HttpOnly flag.

Sample Code

```php
function ierg4210_login(){
  if (empty($_POST['email']) || empty($_POST['pw'])
    || !preg_match("/^[\w=+\-\/][\w='+\-\/\.]*@[\w\-]+(\.[\w\-]+)*(\.[\w]{2,6})$/", $_POST['email'])
    || !preg_match("/^[\w@#$%\^\&\*\-]+$/", $_POST['pw']))
    throw new Exception('Wrong Credentials');

  // Implement the login logic here


  if ($login_success){
    // redirect to admin page
    header('Location: admin.php', true, 302);
    exit();
  } else
    throw new Exception('Wrong Credentials');
}


function ierg4210_logout(){
  // clear the cookies and session

  // redirect to login page after logout
  header('Location: login.php',true,302);
  exit();
}
```

put the code here

What’s next?

- Through login.php
  - We can access the admin panel page and do some operation through admin-process.php
  - What if we directly open ../admin.php?

    (or directly sending request to admin-process.php)
- So, we should validate the token for every time we want to access admin.php
  - Also for admin-process.php
  - If OK, you remain in the admin panel.
  - Otherwise, redirect back to login or main page.

Phase 4.4

- ❑ Create a user table
- ❑ Login.php
- ❑ Maintain an authentication token using Cookies
- ❑ Validate the token
- ❑ Support logout and password changing

Validate the token before revealing and executing admin features

- Validate the authentication token (both admin.php & admin-process.php must validate the auth. token)

```php
function auth(){
  if(!empty($_SESSION['s4210'])) ← Why check SESSION first?
    return $_SESSION['s4210']['em'];
  if(!empty($_COOKIE['s4210'])){
    //stripslashes() Returns a string with backslashes stripped off.
    // (\' becomes ' and so on.)
    if($t = json_decode(stripslashes($_COOKIE['s4210']),true)){
      if (time() > $t['exp']) ← Firstly, check expire or not
        return false; // to expire the user
      $db = newDB();
      $q = $db->prepare('SELECT * FROM account WHERE email = ?');
      $q->execute(array($t['em']));
      if($r=$q->fetch()){
        //expected format: $pw=hash_hmac('sha1', $exp.$PW, $salt);
        $realk=hash_hmac('sha1', $t['exp'].$r['password'], $r['salt']);
        if($realk == $t['k']){
          $_SESSION['s4210'] = $t;
          return $t['em'];
        }
      }
    }
  }
  return false;
}
```

Compute the cookie token again then compare

- Make it as a function in a separated file. (e.g., auth.php) Then include it.

  `include_once('auth.php');`

- Call it firstly in admin.php and admin-process.php.
- If fail, redirect to login.php

Phase 4.4: Other Parts

- Indicate user name on your website mainpage
  - after login or “guest”
  - E.g., using $_SESSION to keep the user name (email)
- Provide logout function
  - In admin.php (maybe also in main.php), a button interface
  - Clear cookie and session in serverside

Sample Code

```php
function ierg4210_login(){
  if (empty($_POST['email']) || empty($_POST['pw'])
    || !preg_match("/^[\w=+\-\/][\w='+\-\/\.]*@[\w\-]+(\.[\w\-]+)*(\.[\w]{2,6})$/", $_POST['email'])
    || !preg_match("/^[\w@#$%\^\&\*\-]+$/", $_POST['pw']))
    throw new Exception('Wrong Credentials');

  // Implement the login logic here


  if ($login_success){
    // redirect to admin page
    header('Location: admin.php', true, 302);
    exit();
  } else
    throw new Exception('Wrong Credentials');
}


function ierg4210_logout(){
  // clear the cookies and session

  // redirect to login page after logout
  header('Location: login.php',true,302);
  exit();
}
```

log out function

Phase 4.4: Points to Note

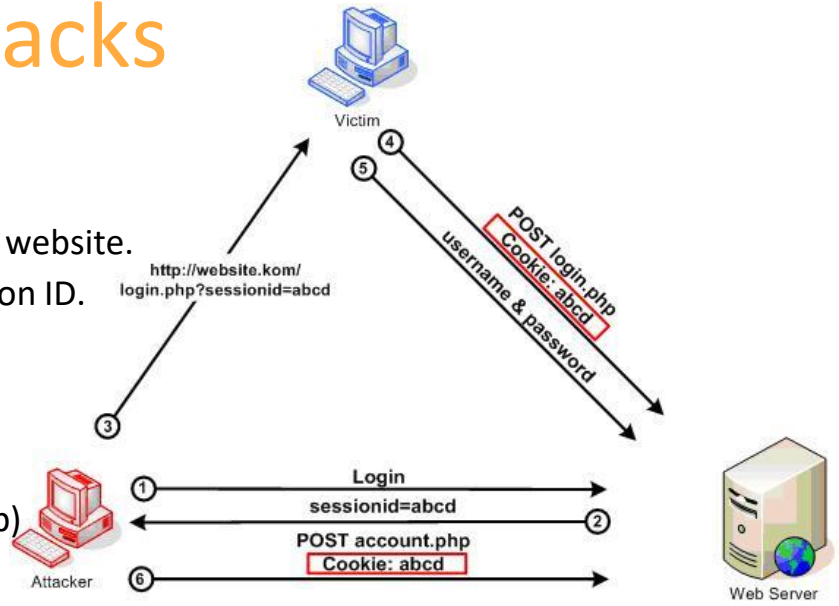
- Indicate user name on your website mainpage
  - after login or “guest”
  - E.g., using $_SESSION to keep the user name (email)
- Provide logout function
  - In admin.php (maybe also in main.php), a button interface
  - Clear cookie and session in serverside
- Support change of Password
  - In login.php, a button interface (input email, old password, new password)
  - In server side (auth-process.php), validate old password first -> update db record -> logout user
- No session fixation vulnerabilities
  - Regenerate the session ID after successful login

    session_regenerate_id( )

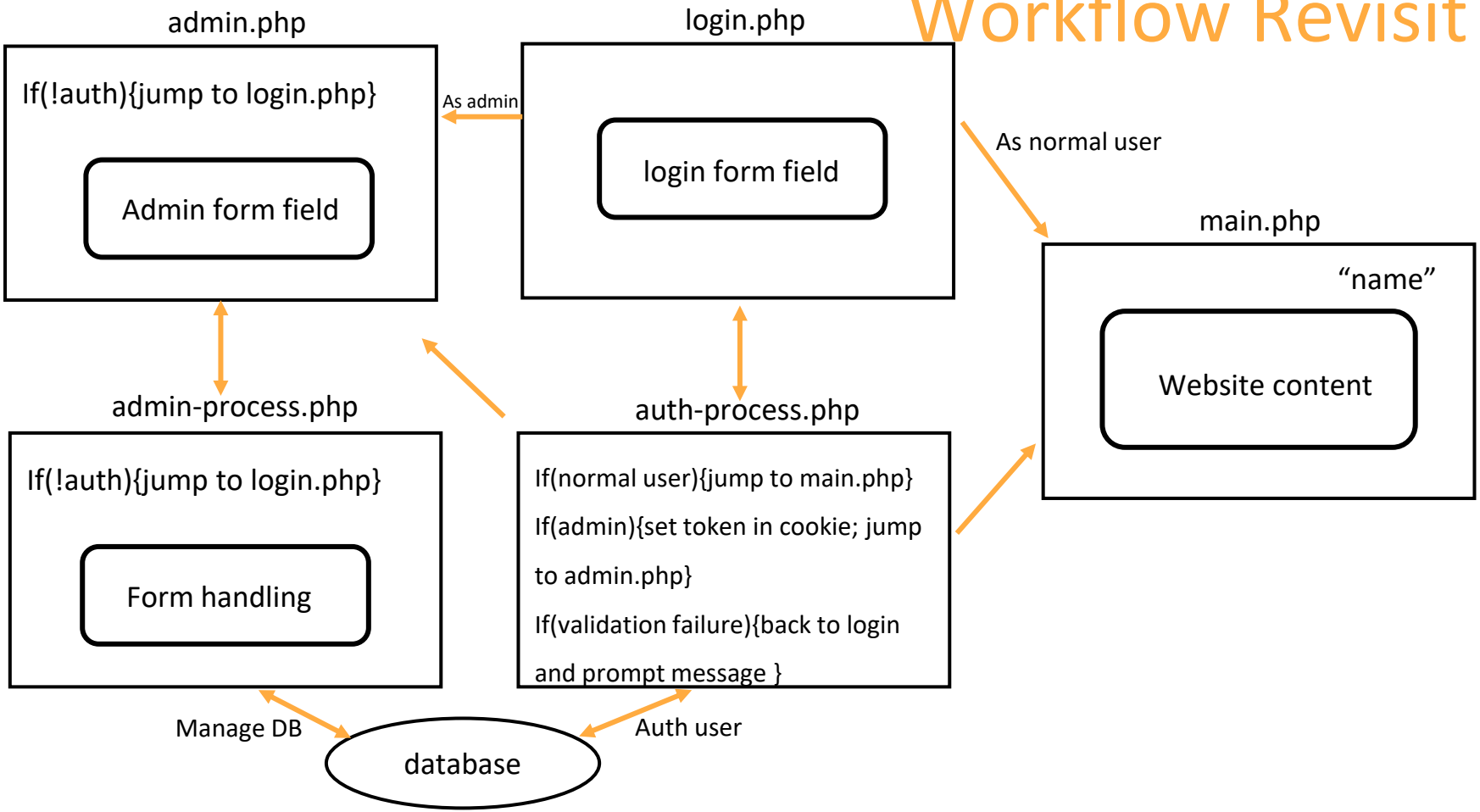
https://www.owasp.org/index.php/Session_fixation

Phase 4.4: Possible Attacks

- Session Fixation Attack
  - First Attacker gets a valid Session ID from the target website.
  - Induce victims to click on a hyperlink with that Session ID.
  - Victim inputs his id and password.
  - Attacker uses that Session ID to do bad things.
- How to prevent?
  - Regenerate session id (session_regenerate_id in php) after user has logged in.
  - Other ways from this picture?
    - http://php.net/manual/en/session.security.php



Workflow Revisit



Reminder

- Watch out the Amazon billing notification
  - May charge you if you open redundant resources
- Secure your private key
- Backup your server data
- Domain names are released. Do NOT hack your classmates' website at this stage!

- Redirect to your main page when accessing the IP or domain name
  - sxx.ierg4210.ie.cuhk.edu.hk → sxx.ierg4210.ie.cuhk.edu.hk/main.php